Attorney Docket No.
224178
MS 193498.01

PATENT APPLICATION

Invention Title:

SCHEMA FOR LOCATION AWARENESS

Inventors:

INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
Yinghua Yao	New Zealand	Bellevue	Washington
Mohammad Shabbir Alam	Pakistan	Redmond	Washington
Vivek Bhanu	India	Bellevue	Washington
Warren V. Barkley	Canada	Mill Creek	Washington
David Buerer	US	Bellevue	Washington
Denise Chen	US	Bellevue	Washington
Florin Teodorescu	Romania	Redmond	Washington
Mark Huyler	US	Kirkland	Washington
John C. Krumm	US	Redmond	Washington
Christopher J. Lang	US	Seattle	Washington
Tim McGrath	US	Redmond	Washington
Timothy M. Moore	UK	Bellevue	Washington

Madhurima Pawar	India	Bellevue	Washington
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
Geoffrey Pease	US	Bothell	Washington
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY
Steven A. N. Shafer	US	Seattle	Washington
INVENTOR'S NAME	CITIZENSHIP	CITY OF RESIDENCE	STATE or FOREIGN COUNTRY

Be it known that the inventors listed above have invented a certain new and useful invention with the title shown above of which the following is a specification.

SCHEMA FOR LOCATION AWARENESS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application relates to co-pending U.S. patent application Ser. No. 10/402,609, filed on March 28, 2003, entitled "Architecture and System for Location Awareness" and naming Mohammad Shabbir Alam, Warren Vincent Barkley, Timothy M. Moore, Geoffrey E. Pease, Steven A. N. Shafer, Florin Teodorescu, Yinghua Yao, Madhurima Pawar and John C. Krumm as inventors, the application being incorporated herein by reference in its entirety.

FIELD OF THE INVENTION

[0002] This invention relates generally to computer systems and, more particularly, relates to a schema for location awareness of computer systems and computer devices.

BACKGROUND OF THE INVENTION

[0003] With the advent of the Internet and the growth of inter-active applications available to computer users comes an increasing need for ubiquitous computing. In this context, ubiquitous computing means the ability of computers to affect most of a user's daily tasks. Computers are called "computers" because of their ability to "compute" or perform mathematical tasks. Computers are no longer seen as only computing machines, but are personal companions that are blending into the fabric of society in the form of personal digital assistants (PDAs) and personal information managers (PIMs), high functioning cellular phones and the like.

[0004] Computers no longer take up the space of an office to be able to compute pi to the 20^{th} decimal place and, instead, fit into the palm of a hand with the same computing

power. Software developers cognizant of the personal companion persona of newer computing tools create user-friendly applications making the computing aspect of computers nearly invisible to users. Such technology results in user interfaces closely resembling human-type interfaces in sharp contrast to prior art computer-readable punch cards required in the past. Another growing technology affecting the ubiquitous nature of computing is wireless technology. Increasing growth of wireless and wired communication networks and the newer types of wireless networks create a need for computers to take advantage of the communication abilities of computers. Wireless no longer means a cellular phone that must be hard-wired within a vehicle, as was known in the past. Modern lithium-ion type batteries and other small but powerful batteries enable cellular phones, PIMS, PDAs and notebook computers to operate for hours at a time without requiring recharging. The long-term operation of computing devices enables a user to move from place to place without concerns of recharging looming while using a device. However, the long-term operation by a user creates opportunities for development of new computing products heretofore not fathomed as being necessary or even possible. One type of new computing product can be referred to as a location awareness product type.

[0005] Current location awareness devices are fragmented, do not work together and are not extensible or unified. For example, global positioning systems, home networking systems, local area networks (LANs) and wireless phones connected to a computing system all are capable of providing data relevant from the location perspective to a computer system. There is no common denominator between these systems allowing synergistic utilization of the location data. Each device outputs location data in different

formats. What is needed, therefore, is a location awareness system that allows for synergy among location awareness products to enhance a user's experience with a computer system and appropriate data structures and schemas to enable such a system to interact with different computer components. Regarding such a schema, what is needed is a schema that operates as an organizational element, enables the functioning for location awareness, and provides a semantic common denominator to promote interoperability.

BRIEF SUMMARY OF THE INVENTION

[0006] Accordingly, a method and schema for a location service includes an abstract location type that contains a variety of location elements. The location elements can include a position, an address, an spatial entity, and an electronic endpoint. The location elements can act as peers that represent a single logical location. The location elements also function as proxies for each other. According to an embodiment, applications which understand one type, but not another, function normally because a resolver populates the specific kind of location information a given application needs based on information determined from one or more other proxies that make up that location object. For example, an application that needs a position can ask a street address geocoder to resolve the address element of the schema, return a position, and store the returned position in the position component of the schema thereby enabling the application that can only understand position. The schema logically wraps disparate location elements that represent a given location

[0007] Thus, the schema supports a location service is extensible and agnostic to the provider of the information and the technology used by the provider. The location service can be either a locally executed module or method or a distributed function that can be aggregated in the cloud.

[0008] A location object can be a collection of elements that defines a location and represents all the information known about any given location. For example, a location object can contain different types of data such as the street address, latitude/longitude, and building/floor/room.

[0009] The schema is configured to be extensible by providing a generic definition of the location element and location object, and by providing the ability to inherit from generic definitions. The schema is also configured to be flexible and can contain zero or more location elements, and each location element can be a location element or anything that extends from it. Applications, services and storage engines can choose to use any subset of the location elements. Location elements can be defined with attributes, such as the city name in a street address. Other location elements can be defined with attributes that include complex data types, referred to herein as nested types. For example, the uncertainty of a position can be described in a nested type. The uncertainty provides information enabling an application/service/storage engine to determine whether position data is accurate and to what extent.

[0010] Additional features and advantages of the invention will be made apparent from the following detailed description of illustrative embodiments, which proceeds with reference to the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] While the appended claims set forth the features of the present invention with particularity, the invention, together with its objects and advantages, can be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

[0012] Figure 1 is a block diagram generally illustrating an exemplary computer system on which the present invention resides;

[0013] Figure 2 is block diagram of an exemplary location aware architecture in accordance with an embodiment of the present invention.

[0014] Figure 3 is a block diagram of a location service within a location aware architecture in accordance with an embodiment of the present invention.

[0015] Figure 4 is a block diagram showing exemplary connections of the location aware system including application programming interfaces in accordance with an embodiment of the present invention.

[0016] Figure 5A is a conceptual diagram describes how the core types relate to each other and how they inherit from the basic WinFS item in accordance with an embodiment of the present invention.

[0017] Figure 5B is a continuation of the conceptual diagram in Figure 5A illustrating core location elements describe indoors and outdoors location information in accordance with an embodiment of the present invention.

[0018] Figure 6 is a conceptual diagram illustrating a more generic implementation of a schema in accordance with an embodiment of the present invention.

[0019] Figure 7 is a conceptual diagram illustrating the nested elements which are used to define attributes of Location Elements and illustrates how they relate and inherit from a WinFS nested type according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0020] Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable computing environment. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0021] Figure 1 illustrates an example of a suitable computing system environment 100 on which the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither

should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0022] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to: personal computers, server computers, hand-held or laptop devices, tablet devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0023] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in local and/or remote computer storage media including memory storage devices.

[0024] With reference to Figure 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of the computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system

components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0025] The computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by the computer 110 and includes both volatile and nonvolatile media, and removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that

has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0026] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136 and program data 137.

[0027] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM,

solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0028] The drives and their associated computer storage media, discussed above and illustrated in Figure 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146 and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers hereto illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a tablet, or electronic digitizer, 164, a microphone 163, a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. The monitor 191 may also be integrated with a touch-screen panel or the like. Note that the monitor and/or touch screen panel can be

physically coupled to a housing in which the computing device 110 is incorporated, such as in a tablet-type personal computer. In addition, computers such as the computing device 110 may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 194 or the like.

[0029] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet. For example, in the present invention, the computer system 110 may comprise the source machine from which data is being migrated, and the remote computer 180 may comprise the destination machine. Note however that source and destination machines need not be connected by a network or any other means, but instead, data may be migrated via any media capable of being written by the source platform and read by the destination platform or platforms.

[0030] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The

modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160 or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0031] In the description that follows, the invention will be described with reference to acts and symbolic representations of operations that are performed by one or more computers, unless indicated otherwise. As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computer of electrical signals representing data in a structured form. This manipulation transforms the data or maintains it at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art. The data structures where data is maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that various of the acts and operation described hereinafter may also be implemented in hardware.

[0032] Referring to Figure 2, an exemplary overview block diagram illustrates an architecture for a location aware service 210. As shown, the location service 210 can be a user mode service 212 and can be installed on a computer, such as computer 110 or on

a handheld computing device. Location service 210 interacts with a plurality of applications 220(1-3) and from outside the device/computer 214. Location service 210 is coupled within the device to kernel mode 215 component such as drivers 222(1-2) and a Windows® file system component 224. The drivers 222(1-2) and Windows® file system component 224 are shown coupled to hardware 216. Each device driver 222(1-2) is shown coupled to its respective hardware device 226(1-2). Windows® file system component 224 is shown coupled to a memory device 230, which could be a database for holding Windows® file system data. Location Service 210 can also be coupled to any compatible remote service, such as a MapPoint® application 240 or other application configured to query for location information and consistent with embodiments herein, or device to provide location data. Location service 210 is also coupled to active directory 260 via lightweight directory access protocol (LDAP) 250. Location Service 210 can also be coupled to an outside device/client 280, which could be coupled to a presence server 270 via a communication channel 214. More particularly, location service 210 could provide location information capable of being retrieved by, for example, a real time communication (RTC) client.

[0033] Referring now to Figure 3, a block diagram illustrates location service 210 in further detail. As shown, the location service 210 includes one or more location application programming interfaces (APIs) 350, a fuser engine 340 and a location management component 330. The location management component 330 interacts with provider plugins 310" and resolvers 320. Location management component 330 can include a user agent 332, a cache 334, a master resolver 336 and a plug-in manager 338. Figure 3 also illustrates providers 310', which can be providers of sensed data that require

at least some interpretation, such as sensor-specific data. For example, a global positioning provider can provide many items of information, including and not limited to items such as time, angle, attitude, a latitude and a longitude. Providers 310' transmit sensed data to provider plugins 310" as device-specific location information or sensor-specific data from devices. Provider plugins 310" can normalize the data, translate the data according to the Location Schema, and transmit the data to location management component 330. Providers 310' can be capable of translating the data prior to transmitting the data to provider plugins 310". The capabilities of the providers 310' are subject to design requirements and limitations. For example, location management component 330 can obtain raw device information from one or more devices via providers 310' or other sources. Location management component 330 normalizes the data, and translates the data into location reports.

[0034] Location management component 330 can be configured within location service 210. Provider plugins 310" operate on the data, however, the plugin manager 338 which is part of the location management system will normalize the data before accepting it and passing it to master resolver 336. Each of the components: master resolver 336, fuser engine 340, and plugin manager 338 indicate readiness to accept information. Thereafter, applications and components that retrieve data do so only when components 336, 338 and 340 have data ready to be retrieved.

[0035] Resolvers 320 receive the sensed and translated data or, in some cases, raw location data, such as device specific location information, and interpret the data. Resolvers 320 can be implemented as plugins to location service 210 and can include device specific location information translators and data sources. Resolvers 320 function

to translate device specific location information to rich location information by using data sources available to infer new data from existing data. In one embodiment, a resolver 320 can translate at least a portion of the data received from one or more devices. Each resolver 320 could be capable of interpreting at least one type of raw or partially-decoded data from a provider 310 and/or be configured to interpret data from another resolver. For example, a resolver 320 could be dedicated to interpreting only 802.11-type data. The resolver would then register with location management component 330 as interpreting only 802.11-type data and would receive only that type of data. Additionally, each resolver 320 can be configured to translate information of another resolver, in which case the resolver might not be capable of translating raw data or partially decoded data, but could also be capable of resolving both raw data, data produced by a provider and data produced by another resover. Figure 3 also shows Windows® file system (WinFS) component 360 that receives data from location service 210.

[0036] Fuser engine 340 generates a current location object. Location service 210 checks current location data via a plug-in or several plugins, including provider plugins 310" and resolver plugins 320. In this context, a plug-in can be implemented as one or more dynamically loadable libraries or dynamic link libraries (DLLs) or other dynamically loadable module capable of expanding the capabilities of software, firmware, or system components. In one embodiment, a notifications application is created on top of a notifications platform for WinFS or other appropriate system that checks to see if there is a match between what an application cares about and the current location. In an embodiment, an assembly contains one or more DLLs; a lower

granularity level of dynamically loadable modules is represented by .NETMODULE files.

System Overview

[0037] Referring now to Figure 4, a block diagram of location system 210 and surrounding components in general illustrates that location service 210 acts as a framework agnostic of any applications, and agnostic as to devices and data sources from which information is obtained. Location service 210 is shown including fuser engine 340, a cache 334, user agent 332, a plug-in manager 338 and master resolver 336. Location service 210 further includes WinFS SQL server 360, shown as an exemplary server only, location provider API 404, location resolver API 414, location notification API 416, location user API 418, and location management API 420. WinFS SQL server 360 couples location service 210 to a notification service 460 and to a user notification API LocUsr 418 and a location notification API 416. APIs 416 and 418 can interact with an application 220.

[0038] Location user API 418 allows an application to query for the current location of the computer running location service 210. Location resolver API 414 specifies what functions need to be implemented by a plugin in order to register with the service. As shown, location user API 418 receives data from WinFS server 360 and passes the data to one or more applications 220(l-n). Location management API 420 receives data from a plurality of components such as fuser engine 340, user agent 332, cache 334, plugin manager 338 and master resolver 336 and passes data back and forth to one or more applications 220. Location management API 420 allows the parameters of the service and components to be configured. Location management API 420 also allows providers

and resolvers to be added and removed. Location notification API 416 receives data from notification service 460 and passes data to applications 220. Location notification API 416 allows an application to register to be notified when the location of the computer running the service has changed. In one embodiment, a separate API, an application registration API allows applications to register for notifications and determining a current location. Applications must first register to obtain location data. Once registered, an application can choose to be notified for predetermined reasons. Location resolver API 414 sends and receives data from plugin manager 412 and master resolver 336 and transmits the data to location resolver user/ Windows® database 408(1), location resolver Active Directory 408(2), and location resolver MapPoint® 408(3), as well as other location resolvers that could benefit from the location data. Master resolver 336 is responsible for managing the resolution of location information. When plugin manager 412 passes to master resolver 336 device-specific location information, master resolver 336 routes the data to the resolvers that are both free and able to resolve the information.

[0039] Specifically, referring to Figure 4, the flow through location service 210 can be described by an example, beginning with providers 402. A provider, for example, an 802.11 provider obtains scan data. If the provider 402 determines that there are new access points, the provider bundles the media access control (MAC) address and signal strength information into an 802.11 report. The 802.11 provider signals that information is ready for plugin manager 338. Plugin manager 338 picks up the sets of information and signals master resolver 336 that there is a new location report to be resolved.

[0040] Next, master resolver 336 retrieves the location report from plugin manager 338. Next, master resolver 336 passes the location report to user agent 332.

[0041] User agent 332 checks cache 334 for this location report. Cache 334 possibly returns a miss. If so, cache 334 caches the data and checks a backend. By checking the backend of cache, a user's location could be determined based on the location report. If cache does not determine the user's location, user agent 332 generates a miss.

[0042] After user agent 332 generates a miss, master resolver 336 passes the location report to a resolver 408, such as active directory (AD) resolver 408(2). AD resolver 408(2) locates the MAC address information in the location report, connects to Active Directory and finds the location of the access point. Next, AD resolver 408(2) returns the location of the access point to master resolver 336 as a location report. Next, master resolver 336 passes the location report to user agent 332. User agent 332 checks cache 334 for a match with the location report returned by AD resolver 408(2). If cache 332 generates a miss, user agent 332 caches the AD report. Next, user agent 332 then checks WinFS 360 to find any saved locations that relate to the location report generated by AD resolver 408(2). If nothing is found, user agent 332 informs master resolver 336 that there is no additional data. Next, master resolver 336 signals to fuser engine 340 that there is data to be retrieved. Fuser engine 340 retrieves two location reports, including an 802.11 location report and an AD location report. Fuser engine 340 fuses these reports and writes a location object representing the fused reports, and both the location reports into WinFS 360 as the current location. Notifications service 460, which can be configured to run on top of WinFS or an equivalent system generates a notification. The generated notification passes through a location notification API and onto the applications registered for the notification.

[0043] Location resolver API 414 is an interface between location service 210 and resolvers. Further, location resolver API 414 allows each resolver to notify the location service 210 that a resolver has new location information. Location resolver API 414 enables the resolver to transfer this information to location service 210. Location provider API 404 sends and receives data from location providers such as providers 402(1-n) which can include an 802.11 provider, a Bluetooth provider, a global positioning system provider and other types of providers of location data. Location provider API 404 is an interface between location service 210 and providers. Location provider API 404 allows each provider to notify the service that it has new location information and transfers this information to the service.

[0044] Location provider API 404 and location resolver API 414 are both part of a plugin manager API. Plugin manager API further includes a plugin manager plugin, which is an interface between plugin manager 338 and a provider plugin 402. The plugin manager plugin interface provides data to both provider and resolver interfaces. Fuser engine 340 functions to fuse data obtained from master resolver 336. Fuser engine 340 generally resolves conflicts and unifies "reports" received from different resolutions of location awareness received from location providers. The data regarding location arrives via master resolver 336 that can filter the data and transmit the data to cache 334 and then to fuser engine 340.

[0045] User agent 332 functions to check cache 334 to determine whether the current location reports may be resolved further using cached data. User agent 332 further checks WinFS 360 to see if the current location reports indicate a location that a user has saved. Cache 334 functions to store resolution trees. Figure 4 illustrates an architecture

that can use location service 210 to provide applications 220 with the ability to query for a current location and the ability to be notified when the location of a user has been changed.

Schema

[0046] Referring now to Figures 5A and 5B, embodiments are directed to a schema for managing data and location aware applications. More particularly, as discussed above, processes within location service 210 enable an instantiation of a location object associated with location service 210 to pass all the location information to an application 220 as an extension of the location object.

[0047] Figures 5A and 5B illustrate data structures that enable the location object to be configured as a generic type of location information component that enables any application to store different or new types of location information into the component and to retrieve data from a common location without having to understand each component of the schema. For purposes of describing the schema, a location is defined as a collection of zero or more location elements each of which represents some type of location information. New location element types may be derived from the basic location element or any of the objects that are derived from it. Nested or helper types are used in location element definitions, and are also extensible themselves. The schema illustrated is a flexible and extensible location schema capable of working with location services, location databases and location aware applications. Additionally, the schema shown can be configured to be stored on devices with small amounts of memory and transferred quickly. Further, the schema is operable with applications 220 and services, such as the location service described above. Location service 210 can use different parts of the

location schema. The schema further can operate with different types of location data. Different types of location data can be stored and integrated into the schema without interfering with or "breaking" existing applications and services. The schema shown is configured to enable a common location schema that enables third parties to provide location information to users, applications and services.

[0048] As shown, the conceptual diagram illustrated in Figures 5A and 5B demonstrates that different types of data can be stored using one schema. Further, applications cooperating with the schema may pick and choose which data to use. The diagram in Figures 5 and 6 illustrates the location object and a collection of location elements it can consist of. An application may understand some of these elements but does not need to understand all, and in particular it may understand some elements in a hierarchy and not their extensions. Thus, for example, Active Directory may be able to understand a MAC address and resolve it into a position which may be understood by a mapping application like MapPoint®. These separate elements can be combined into one object and presented as a multifunctional location object.

[0049] As described above, with reference to Figure 5B, a physical location can be described in many ways: friendly name, address, position, building/floor/room, etc. The schema illustrated in Figure 5A and 5B shows how these descriptions may be accommodated through an extensible data structure.

[0050] The location profile allows applications, services and data stores to associate a location with some context. For example an application could use this associate the printer used at a home where the printer would be the context, the location would be home and the application would be referenced by some identifier. This is represented as

a location profile object. Figure 5A shows how this location object may be implemented in the WinFS schema definition language. The location object and the location element have already been explained in previously. A position is a type of location data. For example, a latitude, longitude pair specifies a location on earth. A coordinate system refers to the space in which the position is located. A possible implementation of the coordinate system object is shown on Figure 5A. We have defined a coordinate system to include units for each of the three dimensions as well as an engineering reference. An engineering reference refers to an entity model. This is used to define new coordinate systems, and in particular can be useful for enterprise administrators that wish to define coordinate systems that describe their floor plans.

[0051] One important aspect of this schema design is that we include device information that can act as a proxy for a location as well as what is traditionally considered to be location data. For example the 802.11 location object includes the MAC address and the signal strength of the access point as perceived by the wireless NIC. This can be used by to infer the user's location because a user must be within a certain proximity to be connected to an access point and see a certain signal strength. Other examples of device information that can act as proxies are IP addresses and phone numbers. Anyone skilled in the art can extend this design to include other devices that have information that can be used to infer the user's location. Figure 5B shows how the 802.11 object can be implemented in WinFS as a IEEE802dot11 object.

[0052] A named location is used to give a name to a location such as "Home", "Work" or "ABC enterprises". Figure 5B shows how this may be implemented in WinFS as a named location object.

[0053] An entity reference is used to refer to entities that can describe locations hierarchically. Each entity reference is a node in this hierarchy that has zero or one parent, zero or more children. The entity reference also has a pointer to the data store and a schema that defines where the entire entity tree may be defined and how it can be accessed. Entities can be used for describing indoors location. For example one entity could be the building, it's child could be the floor, and it's child could be the room. Similarly an entity reference could refer to a reference to an entity that represents a building, that has a child which refers to an entity that represents a floor that has a child which refers to an entity that represents a room. Figure 5B shows how an entity reference may be defined in WinFS. There are many ways to describe hierarchical locations.

[0054] An address refers to the street address of a location. The street address as defined here includes things such as the address line, city and administrative division. Figure 5B shows how this may be implemented in WinFS.

[0055] A position refers to a point in a one, two or three dimensional coordinate space. It can be used to represent the latitude, longitude and altitude of a place on earth. A position has one, two or three scalar values representing the vectors in the coordinate space, a reference to a coordinate system, an uncertainty representing how accurate the position is, and it may have an angle in one, two or three space. Figure 5B shows how this may be implemented in WinFS as a position object.

[0056] Referring now to Figure 6, an embodiment is directed to a generic schema operable with traditional types of location information: address, position, etc, and proxies for location: MAC, IP, phone number and the like. .The location information component

can be configured as an extension to a basic location report that is generic for purposes of allowing any applications to utilize the schema.

[0057] The data flow diagram illustrates that a location schema according to embodiments disclosed herein can be composed of three types of reports, including device reports, logical reports, and position reports. More specifically, a user's location can be described in embodiments herein in at least three ways, including through devices nearby, an actual position and a logical association to an actual user position. The report including each type of location information is shown as a hierarchical report culminating in AReport 610 at the top of the diagram. AReport 610 is a base class for all location elements. All location objects extend from AReport 610 either directly or indirectly.

[0058] One of the three types of reports, the device report, is exemplified in Figure 6 by objects shown as IPReport 616, MACReport 618 and Ieee80211Report 620. The IPReport 616 contains information available to the network layer such as the IP address and the information available to the link layer. MACReport 618 contains information available to the link layer such as the MAC address. Ieee80211Report 620 can be used to describe the signal strength and SSID of an access point as perceived from a user's device.

[0059] Logical location type reports are exemplified by AddressReport 624, BldgFloorRoomReport 622, and NamedLocationReport 626. Each of the AddressReport 624, the BldgFloorRoomReport 622, and NamedLocationReport 626 are provided to base class ACompositeReport 612 as the base class for logical reports. ACompositeReport 612 contains location information that represents a spatial hierarchy. For example, AddressReport 624 contains the country, administrative division, secondary city and

primary city, and addressline. Each of the elements can represent a spatial hierarchy. Additional elements can be added as one of ordinary skill the art with the benefit of this disclosure will appreciate. For example, specific street address information can be included. BuildingFloorRoomReport 622 is also a logical location type of report and can contain building, floor and room information. Additional information can be added by adding elements to NamedLocationReport 626, which allows the user to name a location e.g. "Home" or "Work".

[0060] Each of the hierarchies in the logical report is a HierarchyLevel 614 object. HierarchyLevel object 614 contains the actual content that describes that hierarchy level. For example, content can include elements such as floor number and a spatial certainty parameter that provides a statistical certainty regarding the likelihood that the location service is correct regarding the location of a user relative to a given hierarchy.

[0061] Position type reports are shown as PoseReport 630, VelocitySensorReport 632 and CoodSystem 634. PoseReport 630 describes the position, angle, and coordinate system (via CoodSystem 634) of the user's location. VelocitySensorReport 632 extends from PoseReport 630 to describe the speed at which a user is traveling. Thus, the speed parameter and SpeedUnit provide a context for a position. A direction can be provided by CoordSystem 634, shown as holding elements Code, CodeSpace, Edition, Descriptor, xyUnit, zUnit, and angleUnit. As one of skill in the art with the benefit of this disclosure witll appreciate, the actual elements in each object are exemplary in nature and similar, expanded or reduced elements within each object are included within the general description of each object.

[0062] Figure 7 shows the supporting types that help describe the location but are not location descriptions themselves. Each type feeds into Base.NestedElement 710. Location.Angle3D 720 describes an angle in one, two or three space, as shown by parameters a1:float[1:1], a2:float[0:1] and a3:float[0:1]. Similarly, Location.Matrix 3D 722 is used in the statistical uncertainty type to describe statistical uncertainty in one, two or three space as a 3 by 3 covariance matrix. The covariance matrices for position or angle are the standard way to represent the spread of a Gaussian probability distribution whose mean is the actual position or angle. The Gaussian model represents a probability distribution that smoothly decreases farther from the mean.

[0063] In contrast, Location.SimpleUncertainty 732 represents a sharp cutoff of possible locations or angles at the edge of a sphere, using elements AnglePrecision:float[1:1] and PointPrecision:float[1:1]. For three dimensional locations, a covariance matrix is provided in Location.StatisticalUncertainy 734, which provide the 3x3 covariance matrices of PointCovarianceMatrix or AngleCovarianceMatrix. For two dimensional or one dimensional locations, the matrix can be used by zeroing the entries in the row(s) and column(s) of the matrix that correspond to the unused dimensions.

[0064] Location.Position 3D 724 denotes the vector representing the position in one, two or three space. Location.PositionUncertainty 726 is the abstract base class of a position. Any type for describing position uncertainty can derive from base class 726. Two types of uncertainty are illustrated as deriving from base class 726, including Location.Simple Uncertainty 732 and Location.Statistical.Uncertainty 734. A benefit of separating the types of uncertainty is that it allows developers to have both a simple

solution if a coarse description of uncertainty is good enough, and a rich solution if a detailed description of an uncertainty is required.

[0065] A simple type of uncertainty contains nothing more than a scalar value for angle uncertainty and one for the position uncertainty. A scalar and angle can be used, for example, to denote a radius around the position in which the true position may lie. A statistical uncertainty contains a covariance matrix for angle uncertainty and one for position uncertainty. A matrix can be used to denote an uncertainty distribution that shows the probability of the true position being in a region around a given position. A location report captures the information the location service may tag to a location object such as the creation time. Such as, for example, when the user was at that location and the confidence of the user being in that location. In one embodiment, every location element contains zero or one location reports, shown as, for example, Core.LocationReport 736 to allow any location element generated by the location service to contain the extra information such as confidence and creation time. Core.LocationReport 736 feeds into Core.CategorizedNestedElement 730 as representing a type of nested element. Location.NonScalarString1024 728 is configured to hold nonscalar string values as needed.

[0066] In view of the many possible embodiments to which the principles of this invention can be applied, it will be recognized that the embodiment described herein with respect to the drawing figures is meant to be illustrative only and are not be taken as limiting the scope of invention. For example, those of skill in the art will recognize that the elements of the illustrated embodiment shown in software can be implemented in hardware and vice versa or that the illustrated embodiment can be modified in

arrangement and detail without departing from the spirit of the invention. Therefore, the invention as described herein contemplates all such embodiments as can come within the scope of the following claims and equivalents thereof.